

AD-A079 986

MARYLAND UNIV COLLEGE PARK COMPUTER VISION LAB F/G 9/3

REGION PROPERTY COMPUTATION BY ACTIVE QUADTREE NETWORKS.(U)

NOV 79 T DUBITZKI, A Y WU, A ROSENFELD AFOSR-77-3271

UNCLASSIFIED

TR-823

AFOSR-TR-80-0091

NL

[OF]
ALL
ACT'S 00001

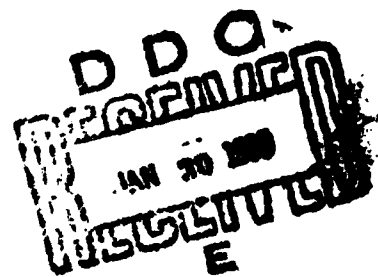
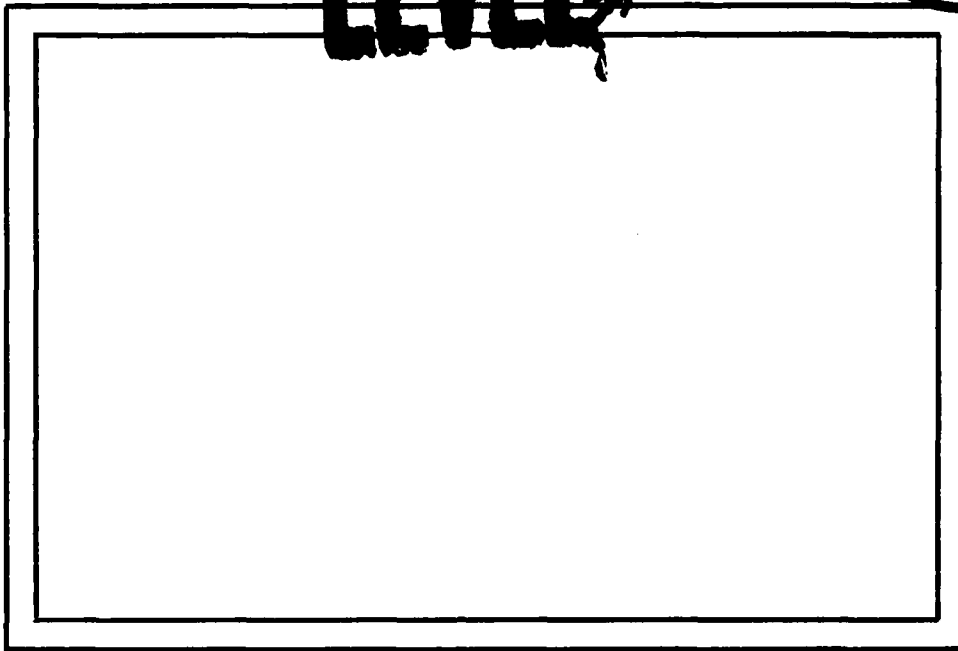


END
DATE
FILMED
2-80
DDC

LEVEL

12

AD A 079988



DDC FILE COPY

UNIVERSITY OF MARYLAND
COMPUTER SCIENCE CENTER

COLLEGE PARK, MARYLAND

20742

Approved for public release;
distribution unlimited.

80 1 29 015

9 [unclear] 42

14 TR-823
15 AFOSR-77-3271

11 November 1979

6 REGION PROPERTY COMPUTATION
BY ACTIVE QUADTREE NETWORKS.

12 25

10 Tsvi/Dubitzki,
Angela Y./Wu
Azriel/Rosenfeld

Computer Vision Laboratory
Computer Science Center
University of Maryland
College Park, MD 20742

DDC
REF ID: A66112
JAN 30 1980
E

ABSTRACT

Given a binary image stored in a cellular array, a local reconfiguration process can be used to reconnect some of the cells into a quadtree network representing the image. This quadtree can also be "roped", i.e., nodes representing adjacent image blocks of the same size can be joined. Using the roped quadtree network, image properties such as perimeter and genus, as well as the quadtree distance transform, can be computed in $O(\text{tree height}) = O(\log \text{ image diameter})$ time. The area and centroid of the image can be computed in $O(\text{height})$ time without the need for roping.

16 [unclear]

18 AFOSR-77-3271

17 A2

This document has been approved
for public release and sale; its
distribution is unlimited.

19 TR 80-2-1

The support of the U.S. Air Force Office of Scientific Research under Grant AFOSR-77-3271 is gratefully acknowledged, as is the help of Kathryn Riley in preparing this paper.

11/1/84

me

1. Introduction

Given a binary image whose pixels are stored at the nodes of an array processor, a tree-structured network of processors representing the quadtree of the binary image is generated by reconfiguration of the array processor based on successive subdivision of array regions which store non-uniform pixel values. (For the details see [1], Section 4.) A roped quadtree is a quadtree whose adjacent blocks in the binary image (WHITE, BLACK, or GRAY) are connected by pointers. In the following sections we describe how an active quadtree can be roped and how this roping helps compute the perimeter of the components in the image or the distance transform of the quadtree, which gives the nearest distance from each BLACK block in the image to the nearest WHITE block border. The ropes are also used to compute the genus, i.e., the number of BLACK components minus the number of holes in them. The area and the centroid of an image stored in an active quadtree are also easily computed.

All these computations need at most $2k$ bits of memory at each processor, where 2^{2k} is the number of nodes in the array processor.

The ropes are needed for checking adjacencies between blocks in constant time. Without ropes, checking adjacencies in parallel may take image diameter time, as many small blocks may try to check the color of a common big neighbor and thus their messages have to pass simultaneously through a common ancestor.

2. Roped quadtrees

A roped quadtree is a quadtree in which nodes corresponding to adjacent blocks (BLACK, WHITE, GRAY) of equal size are connected. These connections are called ropes. Note that only adjacent blocks of equal size are roped, so that each node has bounded degree.

The construction of a quadtree by reconfiguration of the array processor storing a binary image is described in Section 4 of [1]. In essence it is done by successive subdivision of blocks whenever they are found to be not uniformly colored.

A roped quadtree can be constructed in the same way except that whenever a block is subdivided into four quadrants the root node of the father block creates the ropes between its four equal quadrants [2]. Such ropes are also established between adjacent equal size blocks which do not belong to the same father. This is done whenever adjacent roped blocks ("old rope" in Fig. 1), which do not belong to the same father, are both subdivided; their equal-size adjacent sons are told by their fathers to connect themselves by a new rope. The roped quadtree construction takes $O(h)$ time where h is the height of the quadtree.

The ropes can also be established, after the quadtree is constructed, starting from the root and going down level by level. Each father node knows its adjacent blocks via its own ropes, and thus can tell its sons whom they should be roped to.

If there are two unequal adjacent BLACK blocks in a quad-tree then there is a GRAY block which includes the smaller BLACK block and is roped to the bigger BLACK block.

Accession For	
NTIS GEM&I	<input checked="checked" type="checkbox"/>
DDC TAB	<input type="checkbox"/>
Unannounced	<input type="checkbox"/>
Justification	
By _____	
Distribution/	
Availability Codes	
Dist	Avail and/or special
A	

3. Perimeter

Let the image be of size $2^k \times 2^k$ pixels. The perimeter of all the BLACK components in the image represented by a quadtree can be computed in parallel as follows: Starting from the root, each node calculates and stores its level, which is one plus the level of its father, with the root having level 0. Once the level is known, each node knows the total length of its sides, which is 2^{k-i} at level i .

Using a bottom-up process, each GRAY node uses the information from its sons and calculates a 4-tuple (m_1, m_2, m_3, m_4) which gives the total length of the WHITE sections along each of its four borders. Each BLACK node sums its four side lengths, and depending on its roped neighbors, subtracts the following from the sum S :

(1) For each BLACK roped neighbor the length ℓ of a side is subtracted from S , since that side is in the interior of a BLACK component.

(2) For each GRAY roped neighbor (see BLACK node a and GRAY node b in Fig. 2), we subtract $2*(\ell - m_j)$ from S , where m_j is the component of the GRAY neighbor's 4-tuple which corresponds to the common border, so that $\ell - m_j$ is the length of the BLACK borders which are in the interior of a BLACK component. Twice that amount is subtracted, because the small BLACK nodes which are part of the GRAY neighbor do not subtract this common border from their own perimeters. Finally, each BLACK node passes its

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFOSR-TR- 80-0091	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) REGION PROPERTY COMPUTATION BY ACTIVE QUADTREE NETWORKS		5. TYPE OF REPORT & PERIOD COVERED Interim
7. AUTHOR(s) Tsvi Dubitzki Angela Y. Wu Azriel Rosenfeld		6. PERFORMING ORG. REPORT NUMBER TR-823
9. PERFORMING ORGANIZATION NAME AND ADDRESS University of Maryland, Computer Vision Laboratory, Computer Science Center College Park, MD 20742		8. CONTRACT OR GRANT NUMBER(s) AFOSR 77-3271
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Office of Scientific Research/NM Bolling AFB, Washington, DC 20332		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 61102F 2304/A2
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		12. REPORT DATE November 1979
		13. NUMBER OF PAGES 24
		15. SECURITY CLASS. (of this report) UNCLASSIFIED
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
16. DISTRIBUTION STATEMENT (of this Report) Approved for public release; distribution unlimited.		
17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)		
18. SUPPLEMENTARY NOTES		
19. KEY WORDS (Continue on reverse side if necessary and identify by block number) Image processing Pattern recognition Quadtree Cellular processing Parallel processing		
20. ABSTRACT (Continue on reverse side if necessary and identify by block number) Given a binary image stored in a cellular array, a local reconfiguration process can be used to reconnect some of the cells into a quadtree network representing the image. This quadtree can also be "roped", i.e., nodes representing adjacent image blocks of the same size can be joined. Using the roped quadtree network, image properties such as perimeter and genus, as well as the quadtree distance transform, can be computed in $O(\text{tree height}) = O(\log \text{image diameter})$ time, as can the area and centroid even without roping.		

Unclassified

perimeter value up to the root for summing.

Clearly the perimeter computation takes at most $O(h)$ time steps where $h \leq k$ is the height of the quadtree. Augmented memory is needed for the GRAY nodes to store the 4-tuples, and is bounded by k bits; the actual amount needed decreases with the distance of a node from the root.

4. The distance transform

The distance transform of a quadtree is a quadtree in which each BLACK node stores the distance from its center to the nearest WHITE node border. The distance transform can be computed in parallel by an active quadtree.

There are three basic distance measures between pairs of points r, s [3]:

a) Euclidean: $p(r, s) = \sqrt{(x_r - x_s)^2 + (y_r - y_s)^2}$

b) City block: $p(r, s) = |x_r - x_s| + |y_r - y_s|$

c) Checkerboard: $p(r, s) = \max(|x_r - x_s|, |y_r - y_s|)$

The checkerboard distance is chosen to work with here since the region which satisfies $p(r, s) < d$, for constant d , is a square for this measure.

The parallel computation of the distance transform is done as follows: First, each GRAY node computes a 4-tuple giving the distances between the borders of the WHITE descendants closest to its four sides and these sides. (The method of computing this 4-tuple is described at the end of this section.) Then the root of the quadtree sends a signal down to all the BLACK leaves of the tree to compute their smallest distances to a WHITE border. The first ancestral level has at least one WHITE node as a son, otherwise it would not have been split in the quadtree construction process. Thus the distance transform of the BLACK leaves having WHITE brothers is 2^{k-i-1} , where i is the level of these leaves below the root of the tree (see Fig. 3).

Computing the distance transform for lower level (smaller i) black nodes, having only GRAY Or BLACK roped neighbors, is more complicated. Consider Fig. 4 where we have BLACK nodes adjacent (8-adjacency) to GRAY nodes. A BLACK node may have its closest WHITE border either at an adjacent GRAY or WHITE node (8-adjacency) of the same size or at an adjacent WHITE node of larger size. BLACK nodes will be informed about the existence of adjacent WHITE nodes larger than they are by having each WHITE node convey its color (one bit) through its father to all the other BLACK descendants of that father which border its sides and corners. For example, in Fig. 5 the numbered BLACK nodes (1-6) will receive (through their common ancestor) the message that a WHITE node, numbered 7, borders them. Note that in dealing with checkerboard distance we have to deal with 8-adjacency of nodes. If one of the 8-adjacent nodes of the given BLACK node is a WHITE node bigger than it then the distance transform is immediately 2^{k-i-1} (e.g. node 1 in Fig. 5); otherwise, the given BLACK node must have at least one GRAY brother, or it would not have been generated. The BLACK node may have its closest WHITE border in any of its 8-adjacent nodes. Therefore it consults with them via the ropes as follows: For the 4-adjacent GRAY neighbors it checks that component of the 4-tuple belonging to the side adjacent to it; and for the 8-adjacent GRAY neighbors it checks the two components

of the 4-tuple belonging to the sides closest to it (e.g. checking the NW gray neighbor involves checking the eastern and southern components of its 4-tuple). Once the given BLACK node finds out which one of its GRAY neighbors has a WHITE descendant closest to it, it adds its own radius (2^{k-i-1} at level i) to the appropriate component of the 4-tuple of that closest GRAY neighbor and gets its distance transform. The above process of consulting with GRAY neighbors is done sequentially and clockwise: First consulting with a 4-adjacent GRAY node (if any), then with an 8-adjacent GRAY node (if any) roped to the last 4-adjacent node in a clockwise direction, then with another 4-adjacent GRAY node, etc. For example in Fig. 6 the BLACK node 0 checks its GRAY neighbors 1, 2, 3, 4 in sequence.

Upon completion of the distance transform at the nodes below it, a GRAY node passes a signal up to its father. The whole computation terminates when the root of the quadtree gets a completion signal from its four sons. Since the computation of the distance transform starts simultaneously at the BLACK leaves and goes up the tree, it takes $O(k)$ time steps for the process to terminate at the root where $2^k \times 2^k$ is the image size. Clearly a BLACK node having no adjacent WHITE nodes of the same size should wait until its 8-adjacent GRAY neighbors compute their 4-tuples.

Construction of the 4-tuple of distances

First, all the leaves of the quadtree determine the smallest distances from the borders of WHITE nodes inside them to their four sides. Since leaves are either BLACK or WHITE their 4-tuples are immediate: $(0,0,0,0)$ for WHITE nodes and $(2^{\ell+1}, 2^{\ell+1}, 2^{\ell+1}, 2^{\ell+1})$ for BLACK nodes where $\ell=k-i$, i is the level of the leaf in the quadtree, and $2^k \times 2^k$ is the image size. Note that the 4-tuple defined for a BLACK node has no distance meaning but is needed for computing the 4-tuples of its ancestors. When all four sons of a GRAY node notify their father that they have finished computing their 4-tuples, that GRAY node computes its own 4-tuple $(\bar{m}_1, \bar{m}_2, \bar{m}_3, \bar{m}_4)$ as follows:

Suppose $(m_1^{(i)}, m_2^{(i)}, m_3^{(i)}, m_4^{(i)})$ for $i=1,2,3,4$ are the 4-tuples of the four sons of the GRAY node and suppose (1) (2) (3) (4) stand for NE, SE, SW, NW and 1,2,3,4 for North, East, South, and West, respectively (Fig. 7). Then:

$$\bar{m}_i = \min\{m_i^{(\bar{i})}, m_i^{(\bar{i}-1)} + 2^\ell, m_i^{(\bar{i}-2)} + 2^\ell, m_i^{(\bar{i}-3)} + 2^\ell\}$$
$$i = 1,2,3,4$$

where $(\bar{i}-1), (\bar{i}-2), (\bar{i}-3)$ are cyclic subtractions for $i=1,2,3,4$. Here $\ell=k-i$ as above. Augmented memory of size $\ell+1 \leq k$ is needed at the nodes to store these 4-tuples.

5. The Euler number

The Euler number (or genus) of a binary image is the number of connected components minus the number of holes in the image. It is proved in [4] that for a binary image represented by a quadtree the genus is equal to $B-A+S$ where B is the number of BLACK nodes, A is the number of pairs of adjacent BLACK nodes, and S is the number of triples or quadruples of BLACK nodes in the image which surround a point.

In computing the genus each GRAY node in the quadtree should first compute three 4-tuples: One gives the number of BLACK nodes along each of its four sides. The second gives the number of adjacent BLACK nodes in the components along each of its sides. The third indicates whether a GRAY node has BLACK nodes in its four corners. The way these 4-tuples are constructed is described at the end of this section and is done bottom-up during the first phase of the genus computation.

The parallel computation of the genus in an active quadtree is done as follows:

Phase 1: Compute B by sending a message from the root of the quadtree down to the leaves ordering each of the BLACK nodes to send a signal to the root. The root node sums these signals and after $O(k)$ time it has the value of B .

Phase 2: Computing A : The root orders each BLACK node to sense its BLACK or GRAY neighbors. If a roped neighbor along a side of a BLACK node is BLACK we have a pair of equal size

BLACK nodes (ropes exist only between nodes corresponding to equal size blocks). Since each of these BLACK nodes counts the pair (because of symmetry), each one should count only $1/2$ so that the count sums up to 1 pair at a higher level of the tree. If the roped neighbor of a given BLACK node, say a, is GRAY, say b, the GRAY node might have $s \geq 0$ descendant BLACK nodes that are adjacent to node a ($s=2$ in Fig. 8). Thus there are s BLACK pairs formed by node a on one side and the s small BLACK nodes on the other side. The BLACK node a looks for equal size GRAY neighbors along each of its sides and orders them to pass up to their ancestor the number of BLACK pairs generated by their adjacencies with node a, which is given by the appropriate component of the first 4-tuple stored at each GRAY node.

Phase 3: Computing S: We are looking for triples or quadruples of BLACK nodes. Triples may occur along the sides of adjacent nodes (Fig. 9). Quadruples are formed only at a common corner of four BLACK nodes. This phase is divided into two stages: In the first one the number of triples of BLACK nodes which surround a point is computed as in the case of pairs in phase 2 except that now each GRAY node uses its second 4-tuple giving the number of BLACK nodes in each connected component along its sides. Suppose the second 4-tuple looks like $((s_1^1, s_2^1, s_3^1, \dots, s_i^1), (s_1^2, s_2^2, \dots, s_j^2), (s_1^3, s_2^3, s_3^3, \dots, s_k^3), (s_1^4, s_2^4, \dots, s_\ell^4))$ where the superscripts have the meaning 1-North,

2-East, 3-South, 4-West (see Fig. 7). (Storing such a 4-tuple needs augmented memory.) Then whenever a BLACK node senses a GRAY node, say on its northern side, it orders the GRAY node to send up the tree the value $\sum_{n=1}^k (s_n^3 - 1)$, which is the number of triples formed along their common border, plus the number of those triples detected at descendant nodes below it.

In the second stage the number of BLACK quadruples which surround a point in the image is computed. Each one of the BLACK leaves in the quadtree is asked by the root to look for three adjacent BLACK nodes having a common corner with it. This is done by each BLACK node trying to sense its two neighbors in two perpendicular directions, say W and S, and then sense the southern neighbor of the western one which is the western neighbor of the southern one. The sensing process uses the third 4-tuple stored at the neighboring GRAY nodes. The largest GRAY node in the quadruple is the one which counts it since the smaller ones cannot sense the largest one via the ropes. If there are two largest BLACK nodes then each of them counts 1/2; if three largest, 1/3; and if all four at a corner are equal, each one of them counts 1/4.

Computing the three 4-tuples of a GRAY node

The first 4-tuple (s_1, s_2, s_3, s_4) gives the number of BLACK nodes along the node's N, E, S, and W sides correspondingly. It is built bottom up in the quadtree as follows: At the first level above the leaves a GRAY nodes sets $s_1=1$ or 2 if one or two of its northern sons are BLACK, otherwise $s_1=0$. Let $(\bar{s}_1, \bar{s}_2, \bar{s}_3, \bar{s}_4)$ denote the 4-tuple at a father GRAY node. Then

$$\bar{s}_1 = s_1^{NW} + s_1^{NE}$$

$$\bar{s}_2 = s_2^{NE} + s_2^{SE}$$

$$\bar{s}_3 = s_3^{SE} + s_3^{SW}$$

$$\bar{s}_4 = s_4^{SW} + s_4^{NW}$$

The subscripts denote the corresponding sons of the given GRAY father.

The second 4-tuple is also built bottom up: For BLACK nodes the second 4-tuple is $(1,1,1,1)$, while WHITE nodes do not have such a second 4-tuple. At lower levels of the quadtree a combination of the two northern descendants of a GRAY node gives the following "northern" part of the second 4-tuple which indicates the series of numbers of connected BLACK nodes along that GRAY node's northern border (see Fig.10).

Let $((s_1^1, s_2^1, \dots, s_k^1), (s_1^2, \dots), \dots)$ and $(c_1^{NW}, c_2^{NW}, c_3^{NW}, c_4^{NW})$ be the second and third 4-tuples of the NW son of the given GRAY node and let $((\underline{s}_1^1, \underline{s}_2^1, \dots, \underline{s}_\ell^1), (\underline{s}_1^2, \dots), \dots)$ and

$(C_1^{NE}, C_2^{NE}, C_3^{NE}, C_4^{NE})$ be the second and third 4-tuples of the NE sons of that node. Then the "northern" part of the second 4-tuple of the GRAY father is:

$(s_1^1, s_2^1, \dots, s_k^1 + s_1^1, s_2^1, \dots, s_\ell^1)$ if $(C_1^{NE} \text{ and } C_2^{NW}) = 1$
or $(s_1^1, s_2^1, \dots, s_k^1, s_1^1, s_2^1, \dots, s_\ell^1)$ otherwise.

Similar considerations give the second 4-tuple for the other sides of a GRAY node.

The third 4-tuple (C_1, C_2, C_3, C_4) at a GRAY node indicates the existence of BLACK nodes at its corners. BLACK nodes should have $C_1=C_2=C_3=C_4=1$ and WHITE nodes $C_1=C_2=C_3=C_4=0$. At higher ancestral levels a GRAY node has a third 4-tuple $(\bar{C}_1, \bar{C}_2, \bar{C}_3, \bar{C}_4)$ computed by:

$$\bar{C}_1 = C_1^{NW} ; \bar{C}_2 = C_2^{NE} ; \bar{C}_3 = C_3^{SE} ; \bar{C}_4 = C_4^{SW}$$

6. Area

The parallel computation of the area of the BLACK components in the image is done as follows: The root node of the quadtree orders each BLACK node to compute its area and pass it up to the root for summation. Since each BLACK node stores its level i in the quadtree by a top down process, it can compute its area: $2^{(k-i)2}$. Storing area values needs augmented memory of k bits per node. The parallel summation takes at most $O(k)$ time steps for a $2^k \times 2^k$ image. Ropes are not needed for the computation.

7. Centroid

Assume that the root of the quadtree stores the total area of the BLACK components in the image and each BLACK node stores its own area S_i at level i .

The root of the quadtree triggers the computation of the coordinates of each of its BLACK nodes (their geometrical centers) as follows: The root has the coordinates $(0,0)$. It computes the coordinates of each of its BLACK sons by adding or subtracting 2^{k-2} to its own coordinates. Recursively each GRAY node at level i below the root appropriately adds or subtracts 2^{k-2-i} from its own coordinates to compute the coordinates of its BLACK sons. This process terminates at the BLACK leaves of the quadtree. Then starts a bottom-up process of computing the moments of each BLACK leaf by multiplying its coordinates by its area and passing it up to the root of the quadtree. These moments are summed at GRAY nodes on the way up to the root. Upon getting the final sum $(\sum X_i S_i, \sum Y_i S_i)$ the root divides it by the total area of the BLACK components in the image to give the coordinates of the centroid.

Ropes are not needed for the centroid or area computation since there is no interaction between adjacent nodes. The augmented memory needed at the nodes is at most $3k$.

7. Conclusions

Tremendous savings are achieved by parallel computation of the perimeter, distance transform, and genus of an image represented by a quadtree. The parallel computation takes time on the order of log diameter of the image in all these cases. Sequential computation takes log diameter times the number of nodes in the image (for perimeter and distance; see [2] and [5]) or times the number of BLACK nodes in the image (for genus; see [4]). Note that the number of BLACK nodes in the checkerboard case is on the order of the square of the image diameter. The same comparison applies to the computation of the area or centroid of the image.

References

- [1] Tsvi Dubitzki, Angela Wu, Azriel Rosenfeld, Local reconfiguration of networks of processors: arrays, trees, and graphs. TR-790, Computer Science Center, University of Maryland, July 1979.
- [2] G. M. Hunter and K. Steiglitz, Operations on images using quadtrees, IEEE Transactions on Pattern Analysis and Machine Intelligence, April 1979.
- [3] Hanan Samet, A distance transform for images represented by quadtrees. TR-780, Computer Science Department, University of Maryland, July 1979.
- [4] Charles R. Dyer, Computing the Euler number of an image from its quadtree. TR-769, Computer Science Center, University of Maryland, May 1979.
- [5] Hanan Samet, Computing perimeters of images represented by quadtrees. TR-755, Computer Science Department, University of Maryland, April 1979.

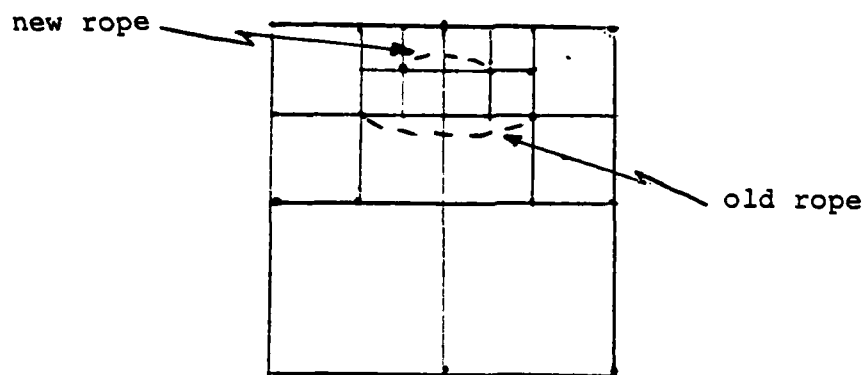


Fig. 1

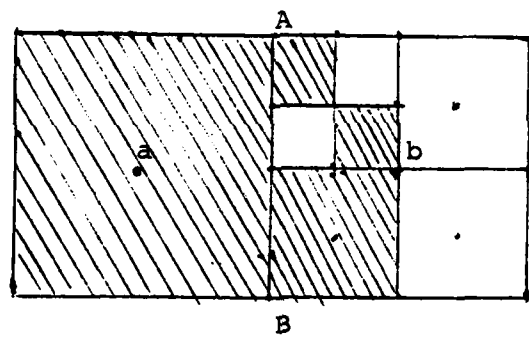


Fig. 2

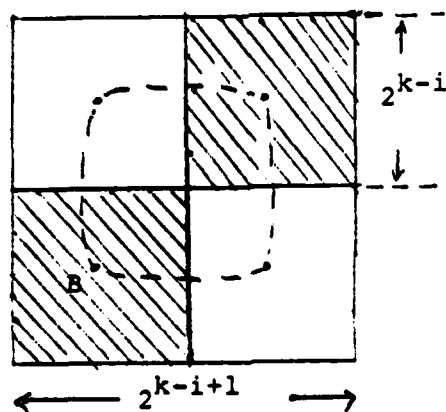
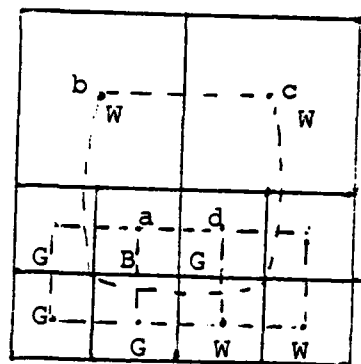


Fig. 3



B--BLACK
 G--GRAY
 W--WHITE
 --- ropes

Fig. 4

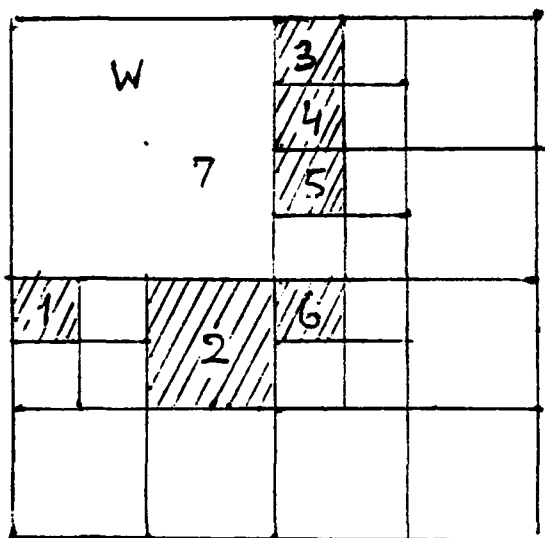


Fig. 5

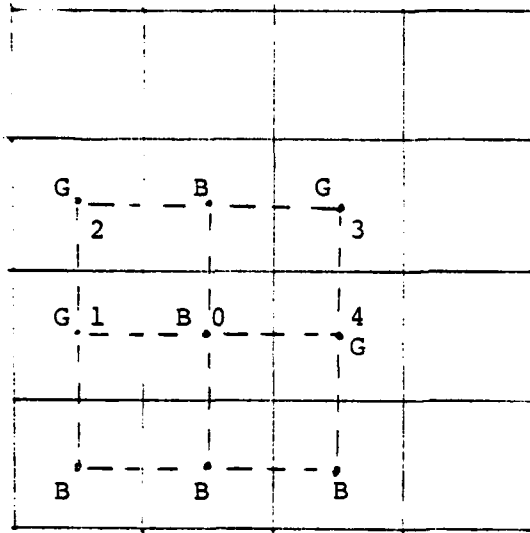


Fig. 6

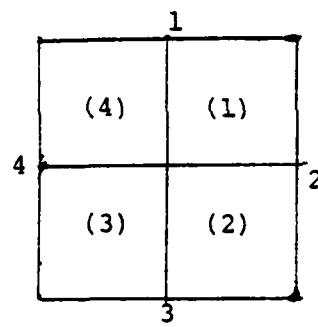


Fig. 7

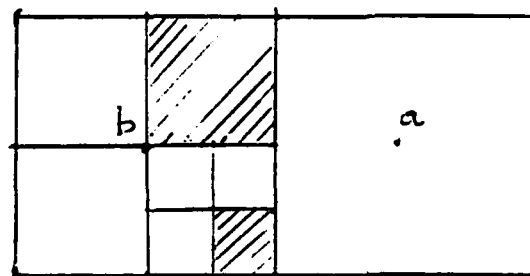


Fig. 8

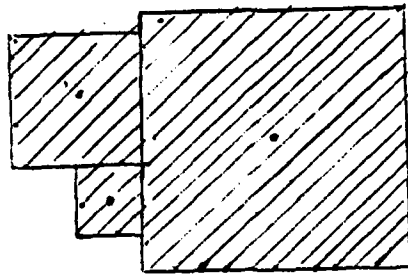


Fig. 9

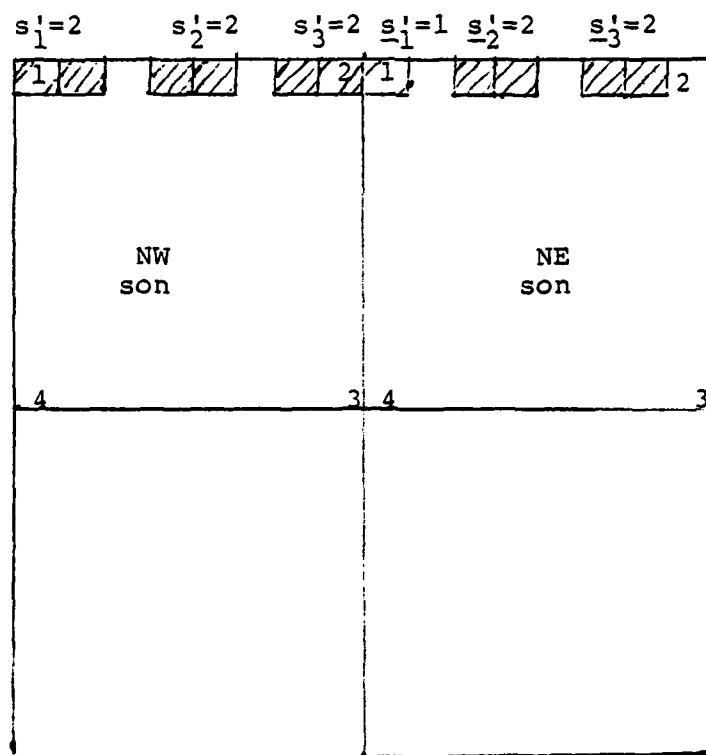


Fig. 10. A GRAY node and its second 4-tuple along the northern border. $C_2^{NW}=1$; $C_1^{NE}=1$.